

On the Correctness of Inverted Index Based Public-Key Searchable Encryption Scheme for Multi-time Search

Shiyu Ji
shiyu@cs.ucsb.edu

Abstract

In this short note we argue that the state-of-art inverted index based public key searchable encryption scheme proposed by Wang et al may *not* be completely correct by giving a counterexample.

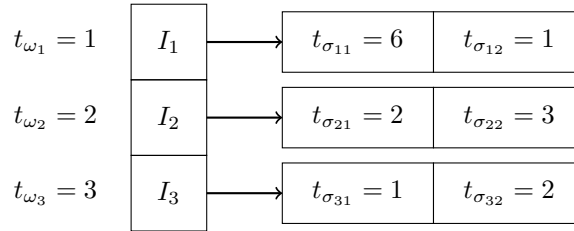
1 Introduction

In INFOCOM 2015, Wang et al [1] proposed the first practical inverted index based public-key searchable encryption scheme with support for multi-time search. In their paper, the correctness of the encryption scheme is argued in Theorem 1, which states their scheme returns all the documents that contain the query keyword(s). We point out if there are more than one keywords in the query, the scheme in [1] is possible to return a document which does not contain all the queried keywords. We show this point by giving a simple counterexample.

2 The Counterexample

In this section all notations and definitions are from [1]. Please refer to [1] for details.

- **Setup**(k): Suppose $k = 4$ (the parameters we choose here are too small to be secure enough, but the correctness should also hold), and the data owner chooses two k -bit primes: $p = 11$ and $q = 13$. Then the data owner generates the key pair (pk, sk) for the Paillier algorithm, in which $n = pq = 143$. The data owner also chooses the following parameters: a pseudorandom permutation f mapping keyword/document identifiers to tags (in integers), and an invertible matrix M . For simplicity, we assume all the keyword/document identifiers are integers. Again, no matter which parameter is chosen, the correctness should be preserved.
- **IndexGen**(M, \mathcal{I}): Suppose the inverted index \mathcal{I} is given as follows:



Here $t_{\sigma_{ij}} = f(\sigma_{ij})$, and $t_{\omega_i} = f(\omega_i) = i$ for each $i \in \{1, 2, 3\}$. Also we have

$$f(\Omega) = \{1, 2, 3\}, \quad f(\Sigma) = \{1, 2, 3, 6\},$$

where Ω is the collection of keywords, and Σ is the collection of documents.

The maximum length of all the inverted lists above is $L = 2$. Since each inverted list has the same length 2, the data owner does not generate any random number for each list. Then the polynomials for the lists can be obtained as follows:

$$P_1(x) = (x - 6)(x - 1) = x^2 - 7x + 6.$$

$$P_2(x) = (x - 2)(x - 3) = x^2 - 5x + 6.$$

$$P_3(x) = (x - 1)(x - 2) = x^2 - 3x + 2.$$

Then the data owner immediately has the polynomial vectors:

$$I_1 = (1, -7, 6)^T.$$

$$I_2 = (1, -5, 6)^T.$$

$$I_3 = (1, -3, 2)^T.$$

Let I be $(I_1, I_2, I_3)^T$. The data owner encrypts the coefficients in I by using Paillier algorithm to get the encrypted index as \tilde{I} . Since $t_{\omega_i} = i$ when i is 1, 2 or 3, we have the dictionary matrix:

$$M_D = \begin{pmatrix} 1 & 8 & 27 \\ 1 & 4 & 9 \\ 1 & 2 & 3 \end{pmatrix}$$

The data owner outsources $M'_D = M \cdot M_D$ and \tilde{I} to the cloud.

- **TrapdoorGen**(M, \mathcal{Q}): Suppose the query is $\mathcal{Q} = \{1, 3\}$, i.e., to query the intersection of I_1 and I_3 . The search user generates two random numbers r_1 and r_2 from \mathbb{Z}_n^* such that neither of them is 1, 2 or 3. We will discuss the choice of r_1, r_2 later. Now the user can get the query polynomial as follows:

$$P'_Q(x) = (x - 2)(x - r_1)(x - r_2) = x^3 - (r_1 + r_2 + 2)x^2 + (r_1r_2 + 2r_1 + 2r_2)x - 2r_1r_2.$$

Thus its coefficient vector is

$$(1, -(r_1 + r_2 + 2), r_1r_2 + 2r_1 + 2r_2, -2r_1r_2).$$

Then the user sends the trapdoor to the cloud:

$$\mathcal{T}_Q = ((1, -(r_1 + r_2 + 2), r_1r_2 + 2r_1 + 2r_2) \cdot M^{-1}, \text{Enc}_{pk}(-2r_1r_2)).$$

- **Query**(\tilde{I}, \mathcal{T}_Q): The cloud server computes

$$\mathcal{V} = \mathcal{T}_Q[1] \cdot M'_D = (1, -(r_1 + r_2 + 2), r_1r_2 + 2r_1 + 2r_2) \cdot M_D.$$

Substituting M_D as before, we have

$$\mathcal{V} = (r_1r_2 + r_1 + r_2 - 1, 2r_1r_2, 3r_1r_2 - 3r_1 - 3r_2 + 9).$$

Then the cloud server computes $v'_i = \text{Enc}_{pk}(v_i) +_h \mathcal{T}_Q[2]$, where $+_h$ denotes homomorphic addition in ciphertexts (Paillier is a partially homomorphic encryption scheme):

$$\mathcal{V}' = (v'_1, v'_2, v'_3) = (\text{Enc}_{pk}(-r_1r_2 + r_1 + r_2 - 1), \text{Enc}_{pk}(0), \text{Enc}_{pk}(r_1r_2 - 3r_1 - 3r_2 + 9)).$$

Finally the server computes $P_R(x) = \mathcal{V}' \cdot \tilde{I}^T$ and sends the polynomial $P_R(x)$ to the search user.¹

¹Here is one doubt: Paillier encryption is *not* fully homomorphic, and hence it cannot support multiplication between two ciphertexts. Thus the encrypted polynomial $P_R(x)$ cannot be directly computed by the cloud server. One possible solution here is: the server sends \mathcal{V}' and \tilde{I} to the user and let the user decrypt them and compute the polynomial $P'_R(x)$ in plaintext.

- **OT**(P_R): Upon receiving $P_R(x)$, the user decrypts the coefficients in $P_R(x)$ and restores the polynomial as follows:

$$P'_R(x) = Dec_{sk}(v'_1) \cdot P_1(x) + Dec_{sk}(v'_3) \cdot P_3(x) = (-2r_1 - 2r_2 + 8)x^2 + (4r_1r_2 + 2r_1 + 2r_2 - 20)x - 4r_1r_2 + 12.$$

Clearly $x = 1$ is one root of $P'_R(x) = 0$. However note that it admits another root:

$$P'_R(x) = 2(x - 1)((-r_1 - r_2 + 4)x + 2r_1r_2 - 6).$$

If $r_1 + r_2 \neq 4$, then another root is

$$x_2 = \frac{2r_1r_2 - 6}{r_1 + r_2 - 4}.$$

Suppose $r_1 = 5$ and $r_2 = 7$, then $x_2 = 8$ will also be admitted as a root (document tag) by the search user. However document tag 8 is not in either I_1 or I_3 . Thus the result is incorrect. In fact, there are many other choices of r_1, r_2 that also give incorrect roots. Here we list a few of them:

r_1	r_2	roots of $P'_R(x)$
4	6	1, 7
6	8	1, 9
7	9	1, 10
5	15	1, 9
9	19	1, 14
9	27	1, 15

Note that since $n = 143$, all the r_1 and r_2 above are in \mathbb{Z}_n^* . All the incorrect roots are integers, and thus the search user has no way to distinguish them as redundant roots.

3 Remarks

The counterexample above is not surprising since given two polynomials P_1 and P_2 , the roots of their sum $P_1 + P_2$ include the common roots of P_1 and P_2 , but may not *only* contain them. In the proof of Theorem 1 in [1], it is not explained why the roots of Eq. (2) are *exactly* the common tags in the queried lists.

If the user only queries one keyword each time, then the scheme in [1] should be correct, since only one list I_i is returned from the cloud server in the form of $s \cdot I_i$, where s is some scalar. There is no summation of polynomials and hence the roots are exactly the tags in I_i .

As a possible way to repair the scheme in [1] for multiple keyword search, the user may query the same keywords for several times, and take the intersection of the returned document tags. However this is more time-consuming, and it is not clear how many query times are enough to guarantee the correctness.

References

- [1] Wang, B., Song, W., Lou, W. and Hou, Y.T., “Inverted index based multi-keyword public-key searchable encryption with strong privacy guarantee”. In 2015 IEEE Conference on Computer Communications (INFOCOM) (pp. 2092-2100).